



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

---

**Институт информационных технологий (ИИТ)  
Кафедра цифровой трансформации (ЦТ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**  
по дисциплине «Разработка баз данных»

**Практическое занятие № 4**

Студенты группы *ИКБО-42-23 Голев С.С.*

\_\_\_\_\_  
(подпись)

Ассистент *Морозов Д.В.*

\_\_\_\_\_  
(подпись)

Отчет представлен «\_\_» \_\_\_\_\_ 2025 г.

Москва 2025 г.

## СОДЕРЖАНИЕ

|   |   |
|---|---|
| СОДЕРЖАНИЕ.....                               | 2 |
| ПОСТАНОВКА ЗАДАЧИ.....                        | 3 |
| ХОД РАБОТЫ.....                               | 4 |
| Использование ранжирующих функций.....        | 4 |
| Использование агрегатных оконных функций..... | 5 |
| Использование функции смещения.....           | 6 |
| Построение сводной таблицы.....               | 7 |
| ЗАКЛЮЧЕНИЕ.....                               | 9 |

## ПОСТАНОВКА ЗАДАЧИ

Задание №1: использование ранжирующих функций.

Для каждой основной «родительской» сущности в вашей БД (например, производитель, категория товара, автор) определить три наиболее значимых по некоторому числовому признаку дочерних сущности (например, три самых дорогих товара, три самые популярные книги по количеству продаж). В результирующей таблице должны быть указаны идентификатор группы, идентификатор дочерней сущности, её числовой признак и ранг. Для расчёта ранга использовать функцию RANK() или DENSE\_RANK().

Задание №2: использование агрегатных оконных функций

Для ключевой сущности, имеющей транзакции по времени (например, товар, услуга), рассчитать нарастающий итог (кумулятивную сумму) по некоторому показателю (например, объем продаж, количество заказов) с разбивкой по временным периодам (месяцам или годам). Отчёт должен содержать идентификатор сущности (id/название/...), временной период, сумму за период и кумулятивную сумму.

Задание №3: использование функции смещения

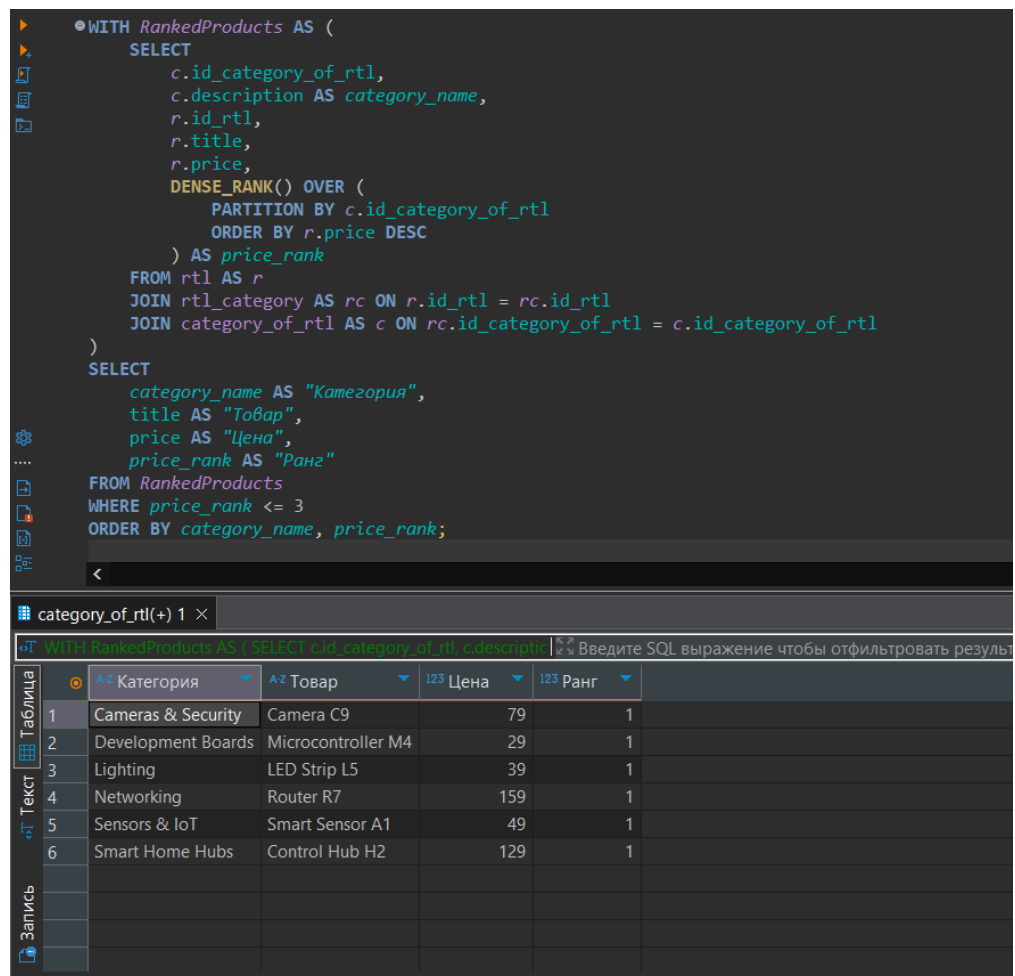
Провести сравнительный анализ общих показателей по периодам. Для каждого периода (например, месяца), начиная со второго, необходимо вывести общий показатель за текущий период и аналогичный показатель за предыдущий период в одной строке. Это позволит наглядно оценить динамику. Необходимо использовать функцию LAG().

Задание №4: построение сводной таблицы

Создать сводный отчет, который агрегирует некоторый числовой показатель для основной сущности по категориям, представленным в виде столбцов.

# ХОД РАБОТЫ

## Использование ранжирующих функций



```
WITH RankedProducts AS (
    SELECT
        c.id_category_of_rtl,
        c.description AS category_name,
        r.id_rtl,
        r.title,
        r.price,
        DENSE_RANK() OVER (
            PARTITION BY c.id_category_of_rtl
            ORDER BY r.price DESC
        ) AS price_rank
    FROM rtl AS r
    JOIN rtl_category AS rc ON r.id_rtl = rc.id_rtl
    JOIN category_of_rtl AS c ON rc.id_category_of_rtl = c.id_category_of_rtl
)
SELECT
    category_name AS "Категория",
    title AS "Товар",
    price AS "Цена",
    price_rank AS "Ранг"
FROM RankedProducts
WHERE price_rank <= 3
ORDER BY category_name, price_rank;
```

|   | Категория          | Товар              | Цена | Ранг |
|---|--------------------|--------------------|------|------|
| 1 | Cameras & Security | Camera C9          | 79   | 1    |
| 2 | Development Boards | Microcontroller M4 | 29   | 1    |
| 3 | Lighting           | LED Strip L5       | 39   | 1    |
| 4 | Networking         | Router R7          | 159  | 1    |
| 5 | Sensors & IoT      | Smart Sensor A1    | 49   | 1    |
| 6 | Smart Home Hubs    | Control Hub H2     | 129  | 1    |
|   |                    |                    |      |      |
|   |                    |                    |      |      |
|   |                    |                    |      |      |

Рисунок 1 – Ранжирующая функция

Для каждой категории (category\_of\_rtl) определяется рейтинг товаров (rtl) по убыванию цены. Используется *DENSE\_RANK()* для учёта одинаковых цен.

## Использование агрегатных оконных функций

The screenshot shows a SQL IDE with a query editor and a results pane. The query uses a window function to calculate a running total of orders by client and month.

```
WITH ClientOrders AS (  
    SELECT  
        c.id_client,  
        c.name AS client_name,  
        DATE_TRUNC('month', TO_DATE(o.date, 'YYYY-MM-DD'))::DATE AS order_month,  
        SUM(o.total) AS monthly_total  
    FROM ordering AS o  
    JOIN client AS c ON o.id_client = c.id_client  
    GROUP BY c.id_client, c.name, order_month  
)  
SELECT  
    client_name AS "Клиент",  
    order_month AS "Месяц",  
    monthly_total AS "Сумма за месяц",  
    SUM(monthly_total) OVER (  
        PARTITION BY client_name  
        ORDER BY order_month  
    ) AS "Нарастающий итог"  
FROM ClientOrders  
ORDER BY client_name, order_month;
```

The results pane displays a table with 6 columns: Client, Month, Sum for month, and Running total. The data is sorted by client name and then by month.

|   | Клиент        | Месяц      | Сумма за месяц | Нарастающий итог |
|---|---------------|------------|----------------|------------------|
| 1 | Alice Johnson | 2025-08-01 | 199,99         | 199,99           |
| 2 | Boris Petrov  | 2025-08-01 | 349,5          | 349,5            |
| 3 | Carla Gómez   | 2025-08-01 | 89             | 89               |
| 4 | David Chen    | 2025-08-01 | 1 299          | 1 299            |
| 5 | Elena Rossi   | 2025-08-01 | 45,75          | 45,75            |
| 6 | Farid Khan    | 2025-08-01 | 599            | 599              |

Рисунок 2 – Агрегатные вычисления в окне

$SUM(...)$   $OVER(PARTITION BY ... ORDER BY ...)$  вычисляет накопительный итог суммы заказов по месяцам для каждого клиента.

## Использование функции смещения

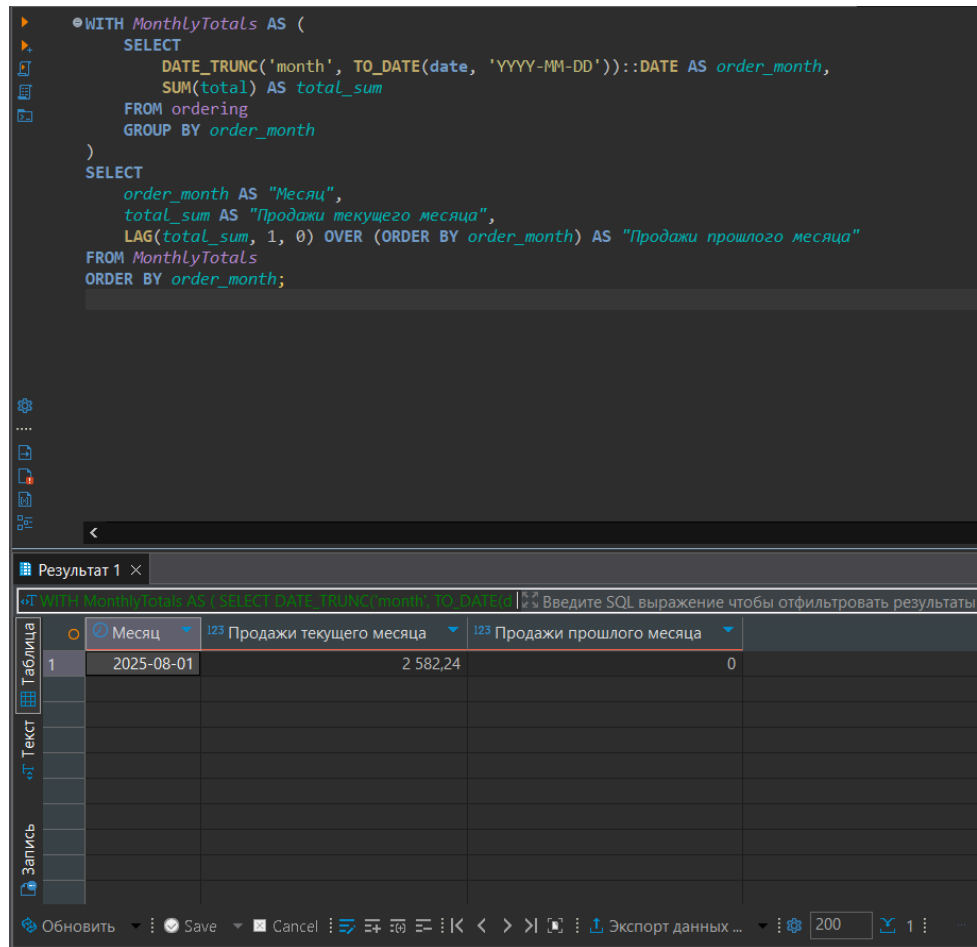


Рисунок 3 – Агрегатные вычисления в окне

Функция  $LAG()$  извлекает значение продаж из предыдущего месяца, чтобы показать динамику роста/спада.

## Построение сводной таблицы

The screenshot displays a database management interface. At the top, a SQL query is entered in a text area. The query uses conditional aggregation to calculate quarterly totals for each employee. Below the query editor, a tab labeled 'employee 1' is active, showing the results of the query in a pivot table view. The table has columns for employee names and four quarters (Q1, Q2, Q3, Q4). The data is sorted by employee name. The interface includes a sidebar with icons for various database operations and a bottom status bar showing the number of rows retrieved and the execution time.

```
SELECT
  e.name AS "Сотрудник",
  SUM(CASE WHEN EXTRACT(QUARTER FROM TO_DATE(o.date, 'YYYY-MM-DD')) = 1 THEN o.total ELSE NULL END) AS "Q1",
  SUM(CASE WHEN EXTRACT(QUARTER FROM TO_DATE(o.date, 'YYYY-MM-DD')) = 2 THEN o.total ELSE NULL END) AS "Q2",
  SUM(CASE WHEN EXTRACT(QUARTER FROM TO_DATE(o.date, 'YYYY-MM-DD')) = 3 THEN o.total ELSE NULL END) AS "Q3",
  SUM(CASE WHEN EXTRACT(QUARTER FROM TO_DATE(o.date, 'YYYY-MM-DD')) = 4 THEN o.total ELSE NULL END) AS "Q4"
FROM ordering AS o
JOIN employee AS e ON o.id_employee = e.id_employee
GROUP BY e.name
ORDER BY e.name;
```

|   | A-Z Сотрудник  | Q1     | Q2     | Q3     | Q4     |
|---|----------------|--------|--------|--------|--------|
| 1 | Artem Sokolov  | [NULL] | [NULL] | 349,5  | [NULL] |
| 2 | Giulia Bianchi | [NULL] | [NULL] | 45,75  | [NULL] |
| 3 | Kenji Tanaka   | [NULL] | [NULL] | 599    | [NULL] |
| 4 | Laura Müller   | [NULL] | [NULL] | 89     | [NULL] |
| 5 | Mika Lahti     | [NULL] | [NULL] | 199,99 | [NULL] |
| 6 | Sarah Smith    | [NULL] | [NULL] | 1 299  | [NULL] |

Обновить Save Cancel 200 6 6 строк получено - 0.098s, 2025-11-11 11:11:11

Рисунок 4 – Агрегатные вычисления в окне

The screenshot shows a SQL IDE with a query window and a results window. The query uses the `crosstab()` function to pivot quarterly order totals for employees. The results window displays a table with 6 rows of employee data and 5 columns for quarters (Q1-Q4).

```

SELECT * FROM crosstab(
    $$
    SELECT
        e.name,
        EXTRACT(QUARTER FROM TO_DATE(o.date, 'YYYY-MM-DD')) AS quarter,
        SUM(o.total)
    FROM ordering AS o
    JOIN employee AS e ON o.id_employee = e.id_employee
    GROUP BY e.name, EXTRACT(QUARTER FROM TO_DATE(o.date, 'YYYY-MM-DD'))
    ORDER BY 1, 2
    $$,
    $$ SELECT q FROM generate_series(1,4) AS q $$
) AS ct("Сотрудник" TEXT, "Q1" NUMERIC, "Q2" NUMERIC, "Q3" NUMERIC, "Q4" NUMERIC);

```

|   | Сотрудник      | Q1     | Q2     | Q3     | Q4     |
|---|----------------|--------|--------|--------|--------|
| 1 | Artem Sokolov  | [NULL] | [NULL] | 349,5  | [NULL] |
| 2 | Giulia Bianchi | [NULL] | [NULL] | 45,75  | [NULL] |
| 3 | Kenji Tanaka   | [NULL] | [NULL] | 599    | [NULL] |
| 4 | Laura Müller   | [NULL] | [NULL] | 89     | [NULL] |
| 5 | Mika Lahti     | [NULL] | [NULL] | 199,99 | [NULL] |
| 6 | Sarah Smith    | [NULL] | [NULL] | 1 299  | [NULL] |

Рисунок 5 – Агрегатные вычисления в окне `crosstab()` превращает строки (кварталы) в столбцы.

Запрос-источник должен возвращать:

- Имя сотрудника;
- Квартал (категорию);
- Значение (сумму заказов).



## ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы были изучены и применены оконные функции SQL, позволяющие проводить аналитические расчёты без потери детализации данных. Были освоены ранжирующие функции для определения лучших элементов внутри групп, агрегатные оконные функции для вычисления нарастающих итогов, а также функции смещения для сравнения показателей по периодам. Кроме того, было реализовано построение сводных таблиц двумя способами — с использованием условной агрегации и функции *crosstab*. Работа позволила закрепить понимание различий между стандартной группировкой *GROUP BY* и оконными вычислениями *OVER()*. Полученные навыки пригодятся при разработке аналитических запросов и формировании отчетов в реальных информационных системах.